Integer Linear Program

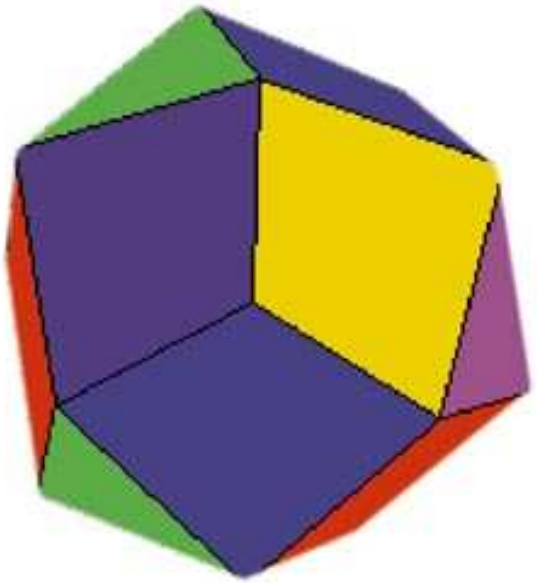
Ein lineares Programm ist ein lin. Ungleichungssystem der Form $A \cdot \underline{y} \leq \underline{b}$ mit $n \times m$–Matrix A und m-dim Vektor b.

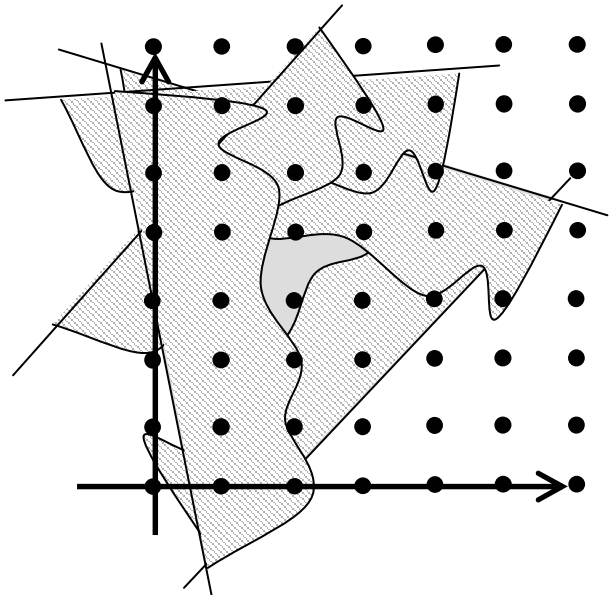
Geometrische Intuition:

$\{\underline{y} \in \mathbb{R}^n \mid \underline{a} \cdot \underline{y} \leq \beta\}$ ist Halbraum,

d.h $\{\underline{y} \in \mathbb{R}^n \mid \underline{a} \cdot \underline{y} \leq \beta\}$ Polytop.

„Enthält es ganz-zahlige Punkte?“





$\mathcal{NP}$-*vollständige Probleme*

SAT $\equiv_p$ **3SAT** $\equiv_p$ **IS** $\equiv_p$ **CLIQUE** $\equiv_p$ **VC** $\equiv_p$ **SubsetSum** $\equiv_p$ **ILP**

$\mathcal{NP}$-vollständige Probleme sind polynomialzeitäquivalent:

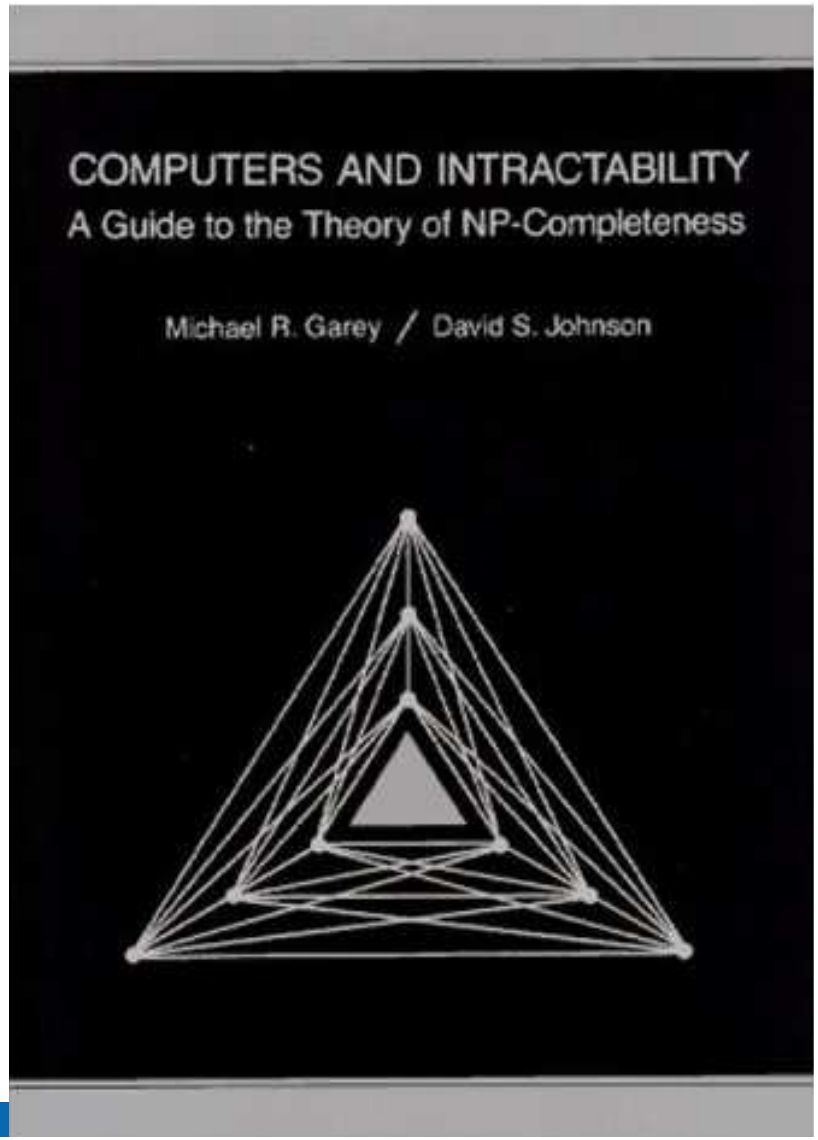
$A, B \in \mathcal{NP}$ und $A \leqslant_p B \leqslant_p A$.

Außerdem $\mathcal{NP}$-vollständig:

- Hamiltonkreis (**HC**),
- Travelling Salesperson (**TSP**)

(Beweise in den Übungen)

- und noch über 300 weitere:



Approximationsalgorithmen

liefern in polynomieller Zeit Lösungen für Optimierungs-probleme, die vom Optimum nur um einen festen Faktor (die ***Güte*** des Appr. Algo) entfernt sind.

TSP: Falls in (G,w) die kürzeste Rundreise Länge k hat, muss ein Appr. Alg. mit Güte c eine Rundreise der Länge $\leq c \cdot k$ liefern.

[Minimierungsproblem, $c > 1$]

Knapsack: Falls $(\underline{g},\underline{w})$ eine Lösung mit Wert w erlaubt, muss ein Approx. Algo mit Güte c eine Lösung mit Wert $\geq c \cdot w$ liefern.

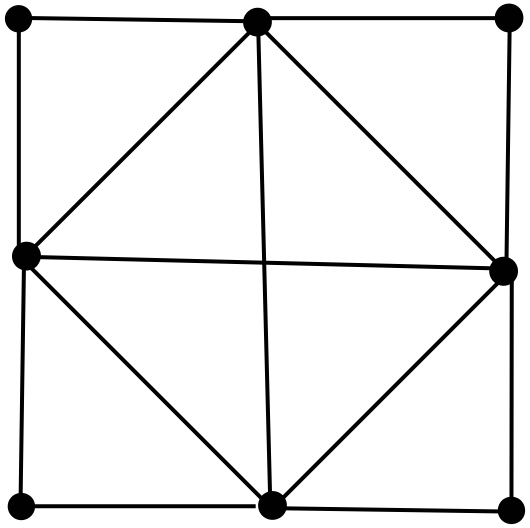
[Maximierungsproblem, $c < 1$]

TSP := { $\langle G,w,k \rangle$ | (G,w) enth. einen Hamiltonkreis mit Gewicht $\leq k$ }

Approximationsalgorithmus für Knotenüberdeckung (VC)

Lemma: (i) Die Knoten eines nicht erweiterbaren Matchings *M* bilden eine Knotenüberdeckung.

(ii) Diese ist höchstens doppelt so groß wie eine optimale.



Beispiele für Matchings:
a) nicht erweiterbar (Größe 2)
b) maximal (Größe 4)

Greedy Algorithmus: Starte mit $M := \emptyset$.

Wähle immer wieder eine Kante $e \in E$, die keinen Knoten mit *M* gemeinsam hat, und setze $M := M \cup \{e\}$.

Approximiert **VertexCover** in Laufzeit $O(|E|)$ mit Güte 2.

Approximationsalgorithmus für das **metrische TSP**

TECHNISCHE UNIVERSITÄT DARMSTADT
Komplexitätstheorie

TSP := { $\langle G,w,k\rangle$ | (G,w) enth. einen Hamiltonkreis mit Gewicht $\leq k$ }

Eingabe: $w : V\times V\to\mathbb{N}$ symmetr. Kantengewichtsfunktion

Gesucht: Rundreise (Permut. π von V) mit min. Gewicht

MTSP Einschränkung: w erfülle Dreiecksungleichung:

$w(a,c) \leq w(a,b) + w(b,c)$ für alle $a,b,c\in V$.

Entscheidungsproblem **MTSP** bleibt $\mathcal{NP}$-vollständig!

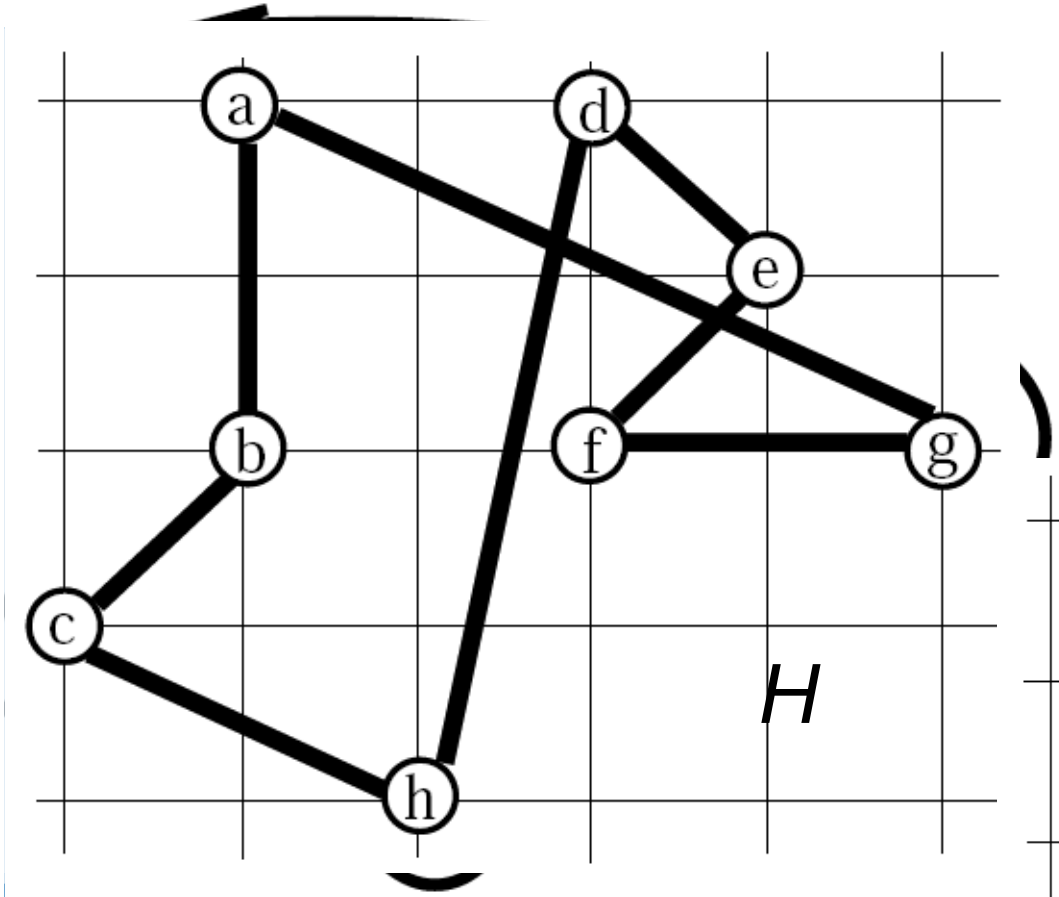
Polynom. Approximationsalgorithmus mit Güte 2:

- Berechne Minimalen Spannbaum T in (G,w).
2. Starte bei belieb. Knoten, durchlaufe T in Preorder.

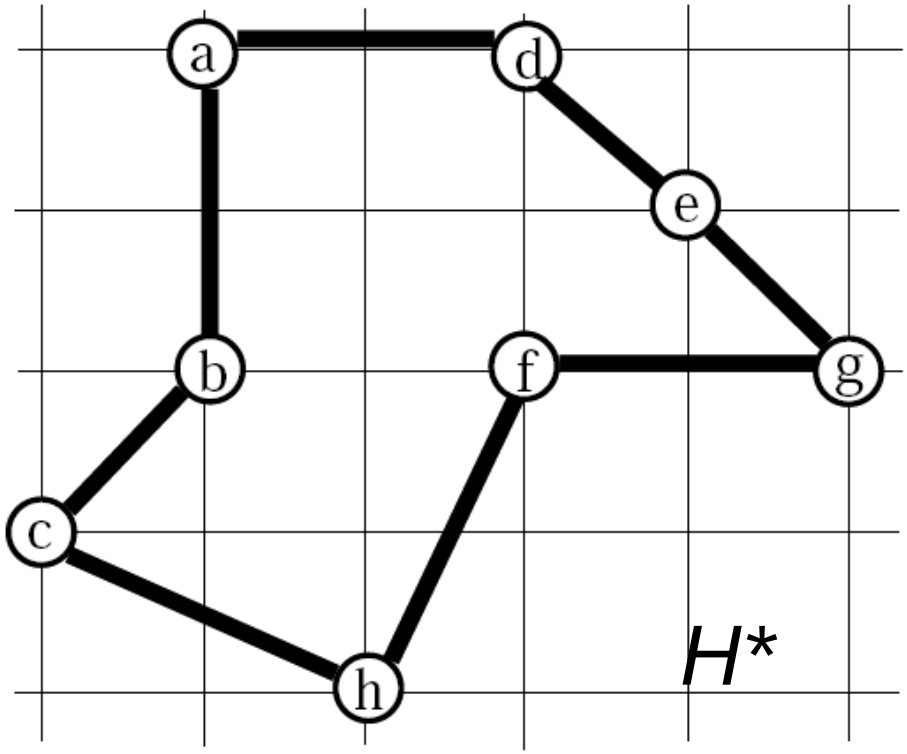
Wurzel links rechts

ETSP zusätzl. Einschränkung: $V \subseteq \mathbb{R}^2$, $w(a,b) := \|a-b\|_2$ ist $\mathcal{NP}$-schwer. Liegt in $\mathcal{NP}$? → Dissertation Prof. Blömer

2. Zähle Knoten von *T* in Preorder (*W*, *L*, *R*) auf



H



*H**

Beweis des Approximationsfaktors 2

TECHNISCHE UNIVERSITÄT DARMSTADT

w erfülle 3ecksungleichung, *T* sei Minimaler Spannbaum. Algorithmus zählt die Knoten von *T* in Preorder auf.

Sei *F* die Folge der in Preorder durchlaufenen Kanten,
H die ausgegebene Tour, *H** eine optimale Tour.

Für Kanten $e_1,\ldots,e_k$ schreibe $L(e_1,\ldots,e_k) := w(e_1)+\ldots+w(e_k)$.

(i) $L(T) \cdot L(H^*)$, da wir aus *H** durch Entfernen irgendeiner Kante einen Spannbaum mit Kosten $\leq L(H^*)$ erzeugen können.

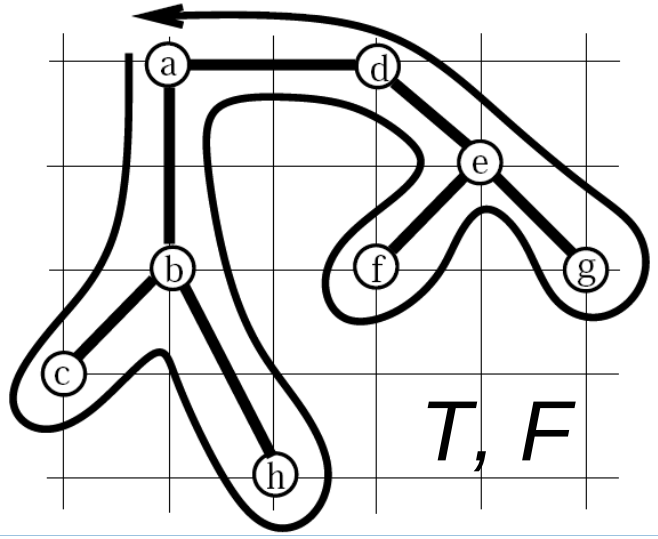
Weil in der Folge *F* jede Kante von *T* genau zwei Mal auftaucht:

(ii) $L(F) = 2 \cdot L(T)$

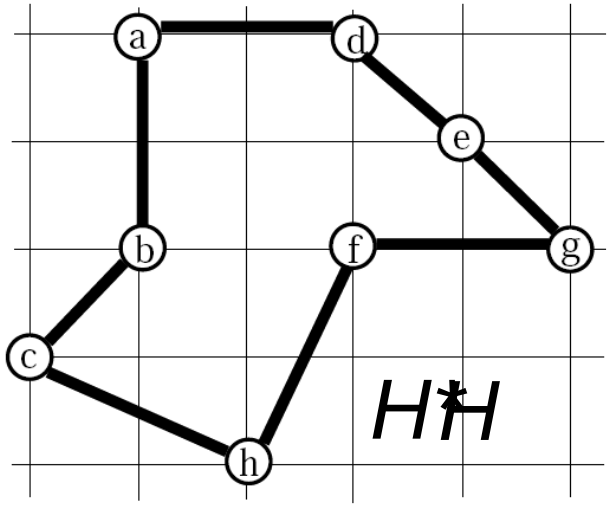
Wegen Dreiecksungl.:

(iii) $L(H) \cdot L(F)$

$\Rightarrow L(H) \leq L(F) = 2 \cdot L(T) \leq 2 \cdot L(H^*)$



T, F

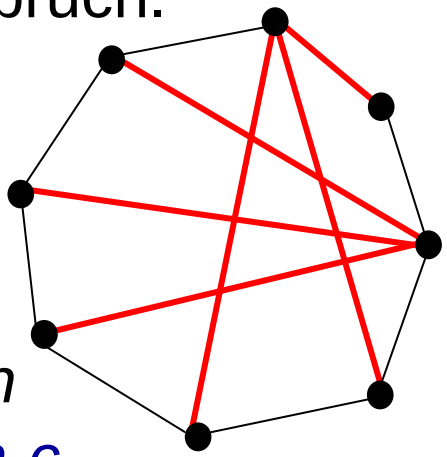


*H**

Grenzen der Approximierbarkeit

Satz: Falls $\mathcal{P} \neq \mathcal{NP}$ gilt,
gibt es kein polynom. Approx.Algo für **TSP** mit konstanter Güte.

Beweis: Wir nehmen an, es gäbe einen polynomiellen Approximationsalgorithmus A für **TSP** mit konstanter Güte $c \in \mathbb{N}$ und entwickeln daraus einen polynomiellen Algorithmus B für **HC**.
Da letzteres $\mathcal{NP}$-vollständig ist, folgt $\mathcal{P}=\mathcal{NP}$: Widerspruch.



Algorithmus B, Eingabe Graph $G=(V,E)$, $n:=|V|$.
Definiere $w(u,v) := 1$ falls $\{u,v\} \in E$;
$w(u,v) := n \cdot c$ falls $\{u,v\} \notin E$.

keine 3ecks-Ungleichung!

$\langle G \rangle \in \mathbf{HC} \Rightarrow$ w enthält Hamiltonkreis mit Gewicht n
$\Rightarrow$ Algo A findet einen mit Gewicht $\leq n \cdot c$
$\langle G \rangle \notin \mathbf{HC} \Rightarrow$ Jeder Hamiltonkreis von w hat Gewicht $\geq n \cdot c + n - 1 > n \cdot c$

HC := { $\langle G \rangle$ | G enthält einen Hamiltonkreis }
TSP := { $\langle G,w,k \rangle$ | (G,w) enth. einen Hamiltonkreis mit Gewicht $\leq k$ }

Knapsack *als Maximierungsproblem*

Gegeben: Werte und Gewichte $w_1,\ldots,w_n,g_1,\ldots,g_n \in \mathbb{N}$ sowie Gewichtsschranke g.

Gesucht: eine Teilmenge $S \subseteq \{1,\ldots,n\}$ mit $\sum_{p\in S} g_p \leq g$, die $\sum_{p\in S} w_p$ maximiert.

Algorithmische Idee: Dynamische Programmierung

Für $S \subseteq \{1,\ldots,n\}$ sei $\mathrm{gew}(S) := \sum_{j\in S} g_j$, $\mathrm{wert}(S) := \sum_{p\in S} w_p$.

Knapsack *als Minimierungsproblem*

Für $S \subseteq \{1,\ldots,n\}$: $\text{gew}(S) := \sum_{p \in S} g_p$, $\text{wert}(S) := \sum_{p \in S} w_p$

a) Suche S mit $\text{gew}(S) \leq g$, das $\text{wert}(S)$ maximiert.

b) Suche S mit $\text{wert}(S) \geq w$, das $\text{gew}(S)$ minimiert.

Sei $\boldsymbol{F_j(w)} := \min \{ \text{gew}(S) \mid S \subseteq \{1,\ldots,j\}, \text{wert}(S) \geq w \}$

kleinstmögliches Gewicht, das mit den ersten j Objekten erzielt werden kann bei Wert mindestens w

Falls kein solches S existiert, sei $F_j(w) := \infty$.

Lemma: i) Der maximale Wert ist $= \max \{ w \mid F_n(w) \leq g \}$

ii) $F_j(w) = 0$ für $w \leq 0$

iii) $F_0(w) = \infty$ für $w > 0$

$F_n(1) \leq F_n(2) \leq F_n(3) \ldots$
$\leq F_n(w) \leq g < F_n(w+1)$

iv) $F_j(w) = \min \{ F_{j-1}(w) , g_j + F_{j-1}(w-w_j) \}$

Exakter Algo für Knapsack

Suche S mit $\text{gew}(S) \le g$, das $\text{wert}(S)$ maximiert

```
LET  w:=0.
WHILE  F_n(w)≤g  DO
        BEGIN
        w := w + 1
        FOR  j:=1…n  DO
                F_j(w) := min { F_{j-1}(w) , g_j + F_{j-1}(w-w_j) }
        END
PRINT w-1
```

Laufzeit $O(n \cdot \text{opt})$

Eingabegröße $\approx$ $\sum_j \log_2(w_j) + \log_2(g_j)$

opt bis zu $\sum_j w_j$

polynomiell nur für „viele kleine“ Pakete

Lemma iv) $F_j(w) = \min \{ F_{j-1}(w) , g_j + F_{j-1}(w-w_j) \}$

$F_n(1) \le F_n(2) \le F_n(3) \le \ldots \le F_n(w) \le g < F_n(w+1)$